JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 25 No. 3, pp. 1-9, March 2020
https://doi.org/10.9708/jksci.2020.25.03.001

Dead Block-Aware Adaptive Write Scheme for MLC STT-MRAM Caches

Seokin Hong*

*Professor, School of Computer Science and Engineering, Kyungpook National University, Daegu, Korea

[Abstract]

In this paper, we propose an efficient adaptive write scheme that improves the performance of write operation in MLC STT-MRAM caches. The key idea of the proposed scheme is to perform the write operation fast if the target MLC STT-MRAM cells contain a dead block. Even if the fast write operation on the MLC STT-MRAM evicts a cache block from the MLC STT-MRAM cells, its performance impact is low if the evicted block is a dead block which is not used in the future. Through experimental evaluation with a memory simulator, we show that the proposed adaptive write scheme improves the performance of the MLC STT-MRAM caches by 17% on average.

▸ **Key words**: STT-MRAM, Cache, Memory, Microprocessor, Dead Block, Simulation

[요 약]

본 논문에서는 MLC STT-MRAM 캐시 메모리의 쓰기 동작 성능을 향상시킬 수 있는 효율적인 쓰기 기법을 제안한다. 제안하는 기법의 핵심 아이디어는 MLC STT-MRAM에 저장된 캐시 블록이 데드 블록 (Dead block)일 경우 쓰기 동작을 빠르게 수행하는 것이다. 이러한 빠른 쓰기 동작은 MLC STT-MRAM에 저장된 캐시 블록을 제거할 수 있지만, 제거된 블록이 앞으로 사용되지 않는 데드 블록일 경우에는 시스템 성능에 미치는 영향이 매우 작다. 메모리 시뮬레이터를 사용한 실험 평가를 통해 본 논문에서 제안하는 쓰기 기법이 MLC STT-MRAM 캐시의 성능을 평균 17% 향상시킬 수 있음을 보인다.

▸ **주제어**: STT-MRAM, 캐시, 메모리, 마이크로프로세서, 데드 블록, 시뮬레이션

• First Author: Seokin Hong, Corresponding Author: Seokin Hong
*Seokin Hong (seokin@knu.ac.kr), School of Computer Science and Engineering, Kyungpook National University
• Received: 2020. 02. 11, Revised: 2020. 02. 25, Accepted: 2020. 03. 03.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

클라우드 서버 및 IoT 장치의 컴퓨팅 요구사항을 충족하려면 제한된 전력 예산 내에서 고성능을 제공하도록 프로세서를 설계해야 한다. 최신 프로세서는 일반적으로 여러 코어를 통합하여 고성능 및 저전력 컴퓨팅 요구사항을 만족시킨다. 프로세서의 코어 수가 증가하고 애플리케이션의 작업 세트 (Working set) 크기가 증가함에 따라 멀티 코어 프로세서에는 더 큰 크기의 캐시 메모리(Cache)가 필요하다. 그러나, 현재 캐시 메모리의 메모리 소자로 사용되는 SRAM (Static Random Access Memory)은 높은 누설 전력 소비 및 낮은 신뢰성으로 인해 확장성에 한계가 있다.

Spin transfer torque magnetic random access memory(STT-MRAM)는 SRAM보다 액세스 시간이 짧고 대기 전력 소모가 적은 새로운 비휘발성 메모리 기술이다 [1]. STT-MRAM은 SRAM보다 셀 크기가 매우 작으므로 캐시 메모리의 데이터 집적도를 크게 향상시킬 수 있다. 이러한 매력적인 특성으로 인해 STT-MRAM은 온칩 캐시 메모리를 위한 차세대 메모리 기술로 상당한 관심을 끌고 있다.

STT-MRAM의 데이터 밀도를 더욱 향상시키기 위해 MLC (Multi-Level Cell) STT-MRAM이 제안되었다 [2]. MLC STT-MRAM은 단일 셀에 두 개의 데이터 비트를 저장함으로써 같은 칩 영역으로 캐시 메모리 용량을 크게 향상시킬 수 있는 장점이 있다 [3, 4, 5, 6, 7, 8, 9, 10]. 하지만 MLC STT-MRAM은 SLC (Single-Level Cell) STT-MRAM보다 읽기 및 쓰기 동작이 느리고 에너지 소모가 큰 단점이 있다. 따라서 MLC STT-MRAM을 캐시 메모리에 사용하기 위해서는 읽기 및 쓰기 동작의 성능을 향상시킬 수 있는 기술이 필요하다.

본 논문에서는 MLC STT-MRAM의 쓰기 동작 성능을 향상시킬 수 있는 데드 블록 탐지 기반의 적응형 쓰기 기법 (Dead Block-Aware Adaptive Write Scheme, DBAW)을 제안한다. DBAW는 앞으로 사용되지 않으리라고 예상하는 캐시 블록 (Cache block)인 데드 블록 (Dead block) [11, 12, 13, 14, 15, 16]을 탐지하여, 만약 MLC STT-MRAM에 데드 블록이 저장되어 있으면, 쓰기 동작을 SLC STT-MRAM 수준으로 빠르게 수행한다. 이때 MLC STT-MRAM에 저장되어 있던 캐시 블록이 삭제될 수 있지만, 그 블록이 데드 블록이라면 캐시 메모리에서 삭제되더라도 성능에 큰 영향을 주지 않는다. 메모리 시뮬레이터를 사용한 성능 평가 실험에서 DBAW 기법은 시스템 성능을 최대 17% 향상시킬 수 있는 것으로 확인되었다.

II. Background

1. STT-MRAM

STT-MRAM은 새로운 비휘발성 메모리 기술로, SRAM에 비해 크기가 매우 작고, 읽기 접근 시간 (Read access time)이 SRAM과 유사한 수준이다 [1]. 또한, 정적 전력 소모 (Static power consumption)가 SRAM보다 매우 적다. 이러한 특성으로 인해 STT-MRAM은 SRAM을 대체하여 대용량 캐시 메모리에 사용될 수 있을 것으로 기대되고 있다.

STT-MRAM은 MTJ (magnetic tunnel junction) 메모리 셀을 데이터 저장 소자로 사용한다. Fig. 1과 같이 MTJ 셀은 2개의 ferromagnetic 층과 oxide barrier (MgO)로 구성되어 있다. 2개의 ferromagnetic 층 중 하나는 고정층 (reference layer)이라고 하며, 자기 방향이 변하지 않는다. 다른 ferromagnetic 층을 자유 층 (free layer)라고 하며, 자유 층의 자기 방향은 MTJ 셀에 새로운 데이터를 기록할 때 달라진다. MTJ 셀은 자유 층의 자기 방향에 따라 다른 저항 상태를 가질 수 있으며, 이는 Fig. 1에서 화살표로 표시되어 있다. 자유층 및 고정층의 자기 방향이 같은 방향 (R0로 표시)일 때는 MTJ의 저항값이 낮고, 두 층의 자기 방향이 서로 다른 방향 (R1으로 표시)일 때는 MTJ의 저항값이 높다.

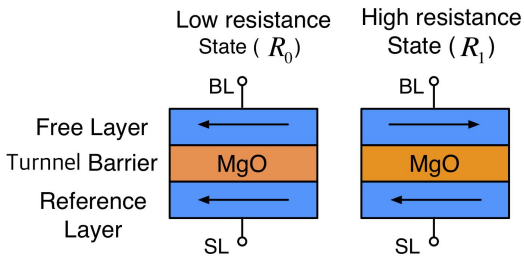


Fig. 1. MTJ

Fig. 2는 액세스 트랜지스터와 MTJ로 구성된 STT-MRAM에 대한 등가회로이다. STT-MRAM에 새로운 값을 기록하기 위해서는 MTJ의 저항 상태를 변경하기에 충분한 시간 동안 MTJ에 스위칭 전류가 주입해야 한다. 예를 들어, MTJ의 저항 상태를 R0로 변경하기 위해서는 고전압을 비트 라인 (BL)에 인가하고 소스 라인 (SL)은 접지에 연결하여 BL에서 SL 방향의 전류(I_w)를 주입한다. MTJ의 저항 상태를 R1으로 변경하려면 전압을 반대로 적용하여 SL에서 BL 방향으로 전류를 주입한다. 읽기 동작도 쓰기 동작과 유사하게 수행된다. 즉, MTJ에

낮은 전류 (I_R)를 주입하고 감지 증폭기 (Sense amplifier)를 사용하여 V_{BL}을 추정함으로써 MTJ의 저항 상태를 측정하여 저장된 값을 읽게 된다.

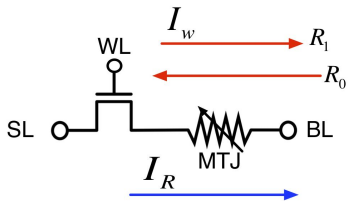


Fig. 2. Equivalent Circuit of STT-MRAM

2. MLC (Multi-Level Cell) STT-MRAM

최근 STT-MRAM의 데이터 밀도를 높이기 위해 두 비트를 저장할 수 있는 MLC (Multi-Level Cell) MTJ가 제안되었다. MLC MTJ는 자유 층을 하드 도메인 (Hard domain)과 소프트 도메인 (Soft domain)으로 나누어 하드 및 소프트 도메인의 자기 방향을 개별적으로 변경함으로써 4가지 다른 저항 상태를 가질 수 있다. Fig. 3과 같이 MLC MTJ는 4가지 저항 상태 (R00, R01, R10, R11)를 갖는다. MTJ는 4가지 저항 상태 중에서 R00 상태의 저항값이 가장 낮고 R11 상태의 저항값이 가장 높다.

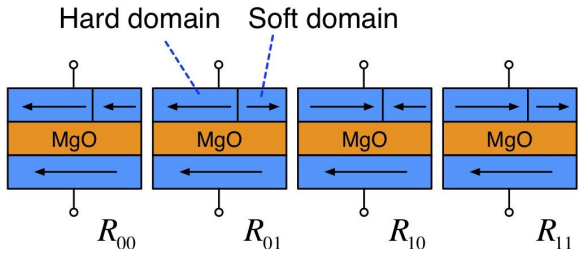


Fig. 3. MLC MTJ

3. MLC STT-MRAM Read and Write Operations

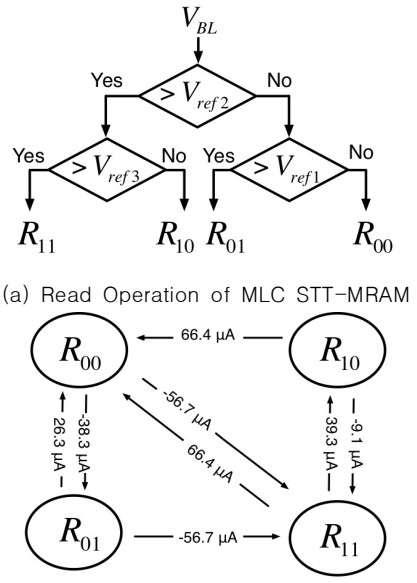
3.1 Read Operation

MLC STT-MRAM의 읽기 동작 (Read Operation)은 이진 탐색 방법 기반으로 수행된다 (Fig. 4a)[2]. 첫 번째 단계에서는 MTJ에 읽기 전류를 주입한 다음, 유도된 V_{BL}을 V_{ref2}와 비교하여 하드 도메인의 자기 방향을 결정한다. 두 번째 단계에서 읽기 전류를 다시 주입한 뒤, 유도된 V_{BL}을 V_{ref3} 또는 V_{ref1}과 비교하여 소프트 도메인의 자기 방향을 결정한다. 따라서, SLC STT-MRAM보다 MLC STT-MRAM의 읽기 지연시간이 약 1.5배 길다 [3].

3.2 Write Operation

MLC STT-MRAM의 쓰기 동작 (Write Operation)은 목표 저항 상태에 따라 다르게 수행된다 [3]. 목표 저항 상태가 R00 또는 R11이면 하드 도메인과 소프트 도메인의 자기 방향이 같으므로 MLC MTJ에 높은 스위칭 전류를 주입하여 하드 도메인과 소프트 도메인의 자기 방향을 동시에 변경한다. 목표 저항 상태가 R01 또는 R10인 경우에는 쓰기 동작이 두 단계로 진행된다. 즉, 첫 번째 단계에서 높은 스위칭 전류를 주입하여 소프트 및 하드 도메인의 자기 방향을 변경시킨 뒤, 두 번째 단계에서는 낮은 스위칭 전류를 주입하여 소프트 도메인의 자기 방향만 변경시킨다. 따라서, 목표 저항 상태가 R01 또는 R10인 경우에는 R00 또는 R11에 비해 쓰기 동작 지연시간이 약 2배 길다 [3].

하드 도메인과 소프트 도메인의 자기 방향을 변경시킬 때 사용되는 스위칭 전류의 크기가 다르다. Fig. 4b에서 볼 수 있는 것과 같이 소프트 도메인에 대한 스위칭 전류가 하드 도메인에 대한 스위칭 전류에 비해 작다.



(a) Read Operation of MLC STT-MRAM

(b) Write Operation of MLC STT-MRAM

Fig. 4. Read and Write Operations of MLC STT-MRAM

4. Data Mapping for MLC STT-MRAM Cache

MLC STT-MRAM을 사용하여 캐시 메모리를 구현하기 위해서는 캐시 블록 (Block)을 MLC STT-MRAM에 사상하는 방법 (Data mapping) 을 결정해야 한다. 가장 기본적인 데이터 사상 방법으로 직접 사상 (Direct mapping, DM) 방법이 있다 (Fig. 5a). 이 방법에서는 하나의 MLC STT-MRAM에 연속된 두 bit를 저장한다. 따라서, 하나의 캐시 블록 (64 Bytes)이 256개의 MLC STT-MRAM에 저

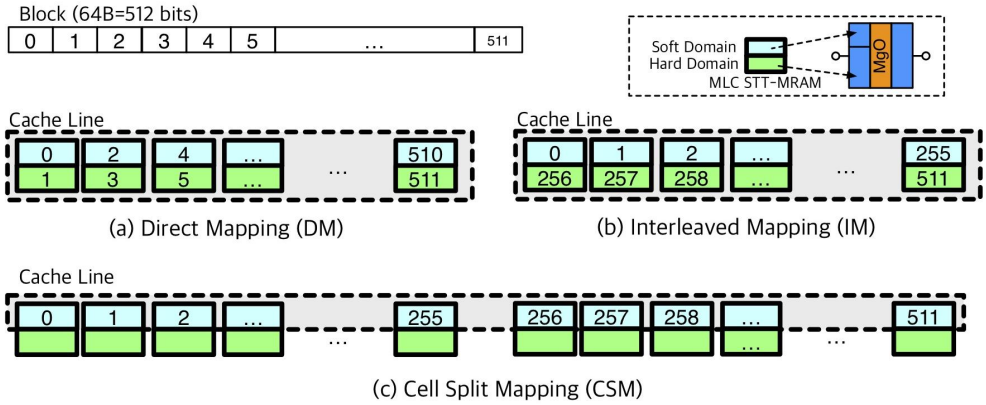


Fig. 5. Data Mapping Schemes in MLC STT-MRAM Cache

장된다. DM 방법은 가장 단순한 데이터 사상 방법이지만 읽기와 쓰기 동작이 오래 걸리는 단점이 있다.

앞서 살펴본 것과 같이 MLC STT-MRAM은 소프트 도메인에 대한 스위칭 전류가 하드 도메인에 대한 스위칭 전류보다 작다. 또한, 쓰기 동작이 소프트 도메인의 자기 방향만 변경한다면 쓰기 지연시간이 SLC STT-MRAM 수준으로 짧아질 수 있다. 이러한 MLC STT-MRAM의 특징을 활용하여 MLC STT-MRAM 캐시 메모리의 성능과 에너지 효율성을 향상시킨 교차 사상 방법 (Interleaved mapping, IM) 방법[5]과 셀 분할 사상 (Cell split mapping, CSM) 방법[4]이 제안되었다.

IM 방법에서는 캐시 블록 (64 Bytes)의 절반 (Half-Block, 32 Bytes)은 하드 도메인에 저장되고 나머지 절반(32 Bytes)은 소프트 도메인에 저장된다 (Fig. 5b). 만약, 소프트 도메인에 저장된 데이터만 업데이트된다면 쓰기 지연시간을 SLC STT-MRAM 수준으로 낮출 수 있는 장점이 있다. 하지만 하드 도메인에 저장된 데이터가 업데이트될 경우는 DM 방법에서와같이 쓰기 동작이 오래 걸리는 단점이 있다.

이러한 IM 방법의 단점을 줄이기 위해 데이터를 절반 크기로 압축하여 소프트 도메인에만 데이터를 쓰도록 하는 기법이 제안되었다 [17]. 이 기법은 데이터 압축이 많이 되는 애플리케이션에 대해서는 좋은 성능을 보일 수 있으나 그렇지 않은 애플리케이션에 대해서는 여전히 DM 방법과 유사한 성능 오버헤드를 갖는 단점이 있다.

IM 방법의 단점을 줄이기 위한 다른 방법으로 가변 블록 크기 (Dynamic block size, DBS) 방법 [6]이 있다. 이 방법에서는 캐시 미스 (Cache miss)가 발생했을 때, 캐시 블록 전부 (64 Byte)를 캐시에 저장하지 않고, 요청된 데이터가 포함된 블록 일부만을 소프트 도메인에 저장한다. 이 기법은 메모리 접근 패턴의 지역성이 낮을 경우에는 효율적일 수 있으나, 지역성이 높을 경우는 캐시 미스 발생률이 크게 증가 될 수 있는 단점이 있다.

CSM 방법에서는 하나의 캐시 블록이 여러 MLC STT-MRAM의 소프트 도메인에만 저장되거나 하드 도메인에만 저장된다 (Fig. 5c). CSM 방법은 캐시 블록이 소프트 도메인에만 저장된다면 쓰기 지연시간이 SLC STT-MRAM 수준으로 짧아지는 장점이 있다. 하지만 캐시 블록이 하드 도메인에 저장된다면 쓰기 지연시간이 DM 방법보다도 더 길어지게 된다. 그 이유는 하드 도메인의 자기 방향을 변경할 때 소프트 도메인의 자기 방향도 함께 변경되어 소프트 도메인에 저장된 캐시 블록이 손실되기 때문이다. 따라서 하드 도메인에 캐시 블록을 저장하기 전에 소프트 도메인에 저장된 캐시 블록을 먼저 읽어둔 뒤, 하드 도메인에 대한 쓰기 동작을 완료한 후에 소프트 도메인에 원래 저장되어 있던 데이터를 복원하는 과정이 필요하다. CSM 방법의 이러한 단점을 최소화하기 위해 빈번하게 업데이트되는 캐시 블록 (Write-intensive block)은 소프트 도메인에 저장하고 그렇지 않은 캐시 블록은 하드 도메인에 저장하거나 [3], 캐시 Associativity 요구 수준이 낮을 경우에는 하드 도메인에 사상된 Way를 사용하지 않도록 하는 기법이 제안되었다 [5]. 하지만 이러한 기법의 효율성은 애플리케이션의 메모리 사용 패턴에 따라 매우 다를 수 있다.

III. The Proposed Scheme

1. Motivation

1.1 Performance Overhead of CSM

본 논문에서는 CSM 방법의 단점을 극복하는 기법을 제안한다. 제안하는 기법에 관해 설명하기에 앞서, 기존에 제

안되었던 CSM 방법의 성능 오버헤드에 대해 분석한다. Fig. 6는 DM 방법이 적용된 MLC STT-MRAM 기반 Last-level Cache (LLC)의 성능 평가 실험결과를 보여준다. 본 실험에는 SPEC CPU2006 벤치마크 중 메모리 집약적인 15개 벤치마크가 사용되었다. RATE 모드에서는 동일한 벤치마크를 다른 프로세서 코어에서 실행하고 MIX 모드에서는 각 프로세서 코어에서 서로 다른 벤치마크를 실행하였다. Fig. 6에서 볼 수 있는 것과 같이, CSM 방법은 DM 방법과 비교하면 RATE 모드에서는 약 7.6% 더 높은 성능을 보여주고 MIX 모드에서는 약 12.6% 높은 성능을 보여주었다. CSM 방법의 최대 성능을 확인하기 위해 모든 메모리 요청에 대한 데이터가 MLC STT-MRAM 기반 LLC의 소프트 도메인에서 사상되어 있는 이상적인 상황 (CSM-Ideal)에 대해 성능을 평가하였다. 실험결과, CSM 방법은 이상적인 상황에서 평균적으로 최대 84% 성능을 향상시킬 수 있음을 확인할 수 있었다.

CSM 방법은 이러한 잠재적 성능향상 효과가 있음에도 불구하고, 하드 도메인에 쓰기 동작을 수행할 때의 매우 높은 성능 오버헤드로 인해 실질적인 성능향상 효과는 매우 작은 것을 알 수 있다. 본 논문에서는 MLC STT-MRAM 캐시 메모리에 대한 CSM 방법의 성능을 극대화하기 위한 기법을 제안한다.

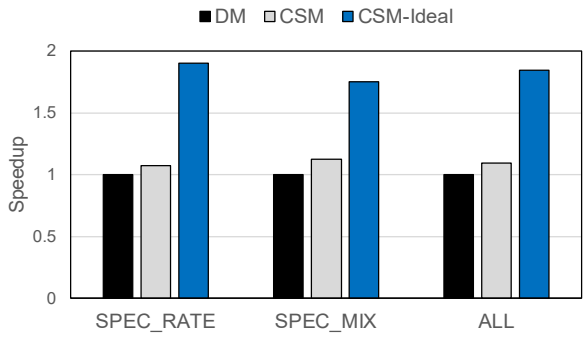


Fig. 6. Performance Comparison of Data Mapping Schemes

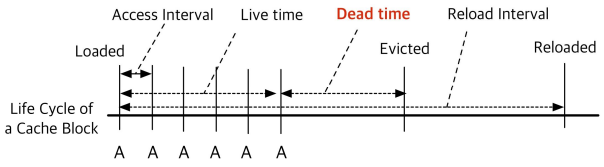


Fig. 7. Life Cycle of Cache Block

1.2 Percentage of Dead Blocks

캐시 메모리에 저장된 캐시 블록 (Cache block) 중, 앞으로 사용되지 않는 캐시 블록을 데드 블록 (Dead block)이라고 한다 [11]. Fig. 7은 캐시 블록이 캐시 메모리에 저장되어 사용되는 패턴 (Life cycle)을 보여준다. 하나의 캐시 블록은 캐시 메모리에 저장된 뒤 (Loaded), 일반적으로 일정한 주기 (Access interval)를 갖고 접근된다. 캐시 블록은 사용이 모두 끝나면 캐시 교체 정책 (Cache replacement policy)에 의해 캐시 메모리에서 제거 (Evict) 되기 전 (Dead time)까지는 그대로 캐시 메모리에 머물게 된다. 이러한 데드 블록이 캐시 메모리 내에 많이 존재하게 된다면 캐시 메모리 공간이 낭비되기 때문에 데드 블록을 찾아 미리 캐시 메모리에서 제거하면 캐시 메모리 성능을 향상시킬 수 있다.

데드 블록 예측은 일반적으로 특정 캐시 블록의 과거 접근 패턴을 학습하여 각 캐시 블록이 데드 블록으로 되는 시점을 예측하는 방법으로 수행된다 [11, 12, 13, 14, 15, 16]. Fig. 8은 시간 기반 예측기 (Time-based predictor) [15]와 계수 기반 예측기 (Counting-based predictor) [14]를 함께 사용하였을 때, LLC에 저장된 캐시 블록 중 데드 블록으로 예측된 블록의 비율을 보여준다. 이 그림에서 볼 수 있듯이 각 캐시 세트 (Set)의 블록 중 평균 45%가 데드 블록으로 예측되었다. Astar 벤치마크의 경우 77% 캐시 블록이 데드 블록으로 예측되었다. 이러한 데드 블록은 앞으로 사용되지 않을 확률이 매우 높으므로 캐시 메모리에서 제거하여도 성능에 큰 영향을 주지 않는다.

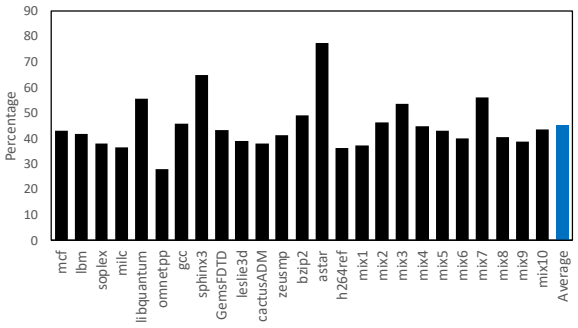


Fig. 8. Dead Block Ratio

2. Dead Block-Aware Adaptive Write (DBAW) Scheme

2.1 Overview

CSM 방법을 사용한 MLC STT-MRAM 캐시 메모리의 성능을 향상시키기 위해 본 논문에서는 데드 블록 탐지 기반의 적응형 쓰기 기법 (Dead Block-Aware Adaptive Write, DBAW) 기법을 제안한다 (Fig. 9). DBAW는 데드 블록 유무에 따라 MLC STT-MRAM의 쓰기 동작을 가변적으로 수행하는 기법이다. CSM 방법에서는 하드 도메인에 사상된 캐시 블록을 업데이트할 때, 한 번의 읽기 동작과 두 번의 쓰기 동작을 수행하게 된

다. 그 이유는 하드 도메인의 자기 방향을 변경할 때 소프트 도메인의 자기 방향도 함께 변경되기 때문에, 하드 도메인에 저장된 캐시 블록을 업데이트한 뒤, 소프트 도메인에 저장되어 있던 캐시 블록을 복원해 줘야 하기 때문이다. 만약, 소프트 도메인에 저장된 캐시 블록이 데드 블록일 경우, 하드 도메인에 저장된 캐시 블록을 업데이트한 뒤 소프트 도메인을 복원해 줄 필요가 없다. 그 이유는 데드 블록은 더는 사용되지 않는 캐시 블록이기 때문이다. 따라서, DBAW 기법에서는 쓰기 동작이 하드 도메인에 저장된 캐시 블록을 업데이트할 때, 만약 소프트 도메인에 저장된 캐시 블록이 데드 블록일 경우에는 SLC 수준으로 빠르게 쓰기 동작 (Fast write)을 수행한다. 이 때 소프트 도메인에 저장되어 있던 캐시 블록은 삭제되기 때문에 해당 캐시 라인 (Cache line) 상태를 비활성 (Invalid) 상태로 변경한다.

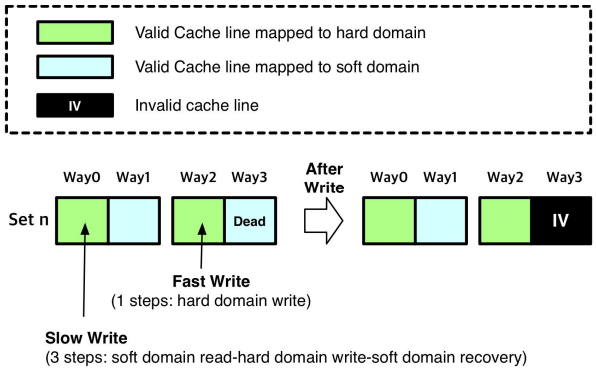


Fig. 9. DBAW Overview

2.2 Dead Block Detection

DBAW 기법에서는 소프트 도메인에 저장된 캐시 블록이 데드 블록인지를 예측해야 한다. 이를 위해 본 연구에서는 계수 기반 예측기 [14]와 시간 기반 예측기 [15]를 함께 사용하여 데드 블록을 예측하였다. 시간 기반 예측기는 캐시 블록이 캐시 메모리에 저장되어 사용되는 시간을 학습하고 이 시간의 두 배 동안 해당 캐시 블록이 접근되지 않으면 그 블록을 데드 블록으로 예측한다. 계수 기반 예측기는 특정 캐시 블록이 캐시 메모리에서 제거되기 전까지 접근되었던 횟수를 기록한다. 해당 블록이 다시 캐시 메모리에 저장되어 사용될 때, 그 캐시 블록의 이전 접근 횟수를 사용하여 그 블록이 데드 블록으로 되는 시점을 예측한다.

시간 기반 예측기를 구현하기 위해 각 태그 엔트리 (Tag entry)에는 두 개의 카운터 (Time counter, Access counter)와 두 개의 레지스터 (Time register, Access counter)가 추가된다. 시간 카운터 (Time counter)는 캐시 블록의 활성 시간 (Live time) (Fig. 7)을 측정하기 위해 사용된다. 이 카운터 값은 특정 시간 간격 (Global tick)으로 증가한다. 접근 카운터 (Access counter)는 캐시 블록의 접근 횟수(Fig. 7)를 측정하기 위해 사용된다. 해당 캐시 블록에 대해 읽기 또는 쓰기 동작이 수행될 때마다 접근 카운터 값은 증가한다. 시간 카운터와 접근 카운터는 모두 포화 카운터 (Saturating Counter)이기 때문에 최대값에 도달하면 더는 증가하지 않는다.

캐시 블록이 캐시 메모리에서 제거될 때, 해당 캐시 블록에 대한 카운터 값도 캐시 메모리에서 삭제된다. 따라서 시간 카운터와 접근 카운터 값도 메인 메모리에 저장한다. 캐시 블록이 캐시 메모리에 다시 저장될 때 (Reload), 메인 메모리에 저장되어 있던 시간 카운터와 접근 카운터 값을 읽어 시간 레지스터와 접근 레지스터에 저장한다. 이 두 레지스터에 저장된 값은 해당 캐시 블록이 데드 블록으로 되는 시점을 예측할 때 사용된다.

2.3 Proposed Write Scheme

Fig. 10은 CSM 방법을 적용한 MLC STT-MRAM 캐시 메모리에서의 DBAW 동작을 설명한 순서도이다. 쓰기 요청(Write request)의 대상이 되는 캐시 블록이 소프트 도메인에 저장되어 있다면 기존 CSM 방법에서처럼 소프트 도메인의 자기 방향만 변경하기 때문에, 쓰기 동작을 빠르게 수행한다 (Fast write). 쓰기 요청에 대한 캐시 블록이 하드 도메인에 저장되어 있으면, 쓰기 동작은 소프트 도메인에 저장된 캐시 블록의 상태에 따라 다르게 수행된다. 만약, 소프트 도메인에 유효한 캐시 블록이 저장되어 있지 않거나 (Invalid) 저장된 캐시 블록이 데드 블록 상태라면 쓰기 동작을 빠르게 수행한다 (Fast write). 만약 소프트 도메인에 유효한 캐시 블록이 저장되어 있고 그 블록의 상태가 데드 블록이 아닌 경우에는 3단계 쓰기 동작 (Slow write)을 수행한다. 즉, 소프트 도메인에 저장된 캐시 블록을 읽은 후, 하드 도메인에 저장된 캐시 블록을 업데이트한다. 마지막으로 원래 소프트 도메인에 저장된 캐시 블록을 다시 소프트 도메인에 저장한다. 이러한 3단계 쓰기 동작은 시간이 오래 걸리고 쓰기 동작을 수행 중인 캐시 뱅크 (bank)에 대해서는 새로운 읽기 요청 (Read request)를 처리할 수 없으므로 성능을 매우 감소시킬 수 있다. 따라서 3단계 쓰기 동작의 수행 횟수를 최소화할 수 있도록 데드 블록 예측을 정확하게 수행해야 한다.

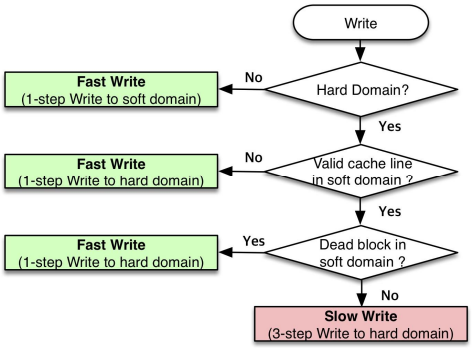


Fig. 10. Write Operation of MLC STT-MRAM Cache with DBAW

IV. Experiments

1. Experimental Methodology

DBAW의 성능 이점을 평가하기 위해 USIMM [18] 메모리 시뮬레이터를 기반으로 메모리 시스템 시뮬레이터를 개발하였다. USIMM을 확장하여 프로세서 코어 및 캐시 메모리를 모델링하였다. 프로세서는 비순차적 실행 (Out of Order Execution)을 모델링하였다. 캐시 메모리 모델은 LRU, DRRIP 및 DIP와 같은 다양한 캐시 교체 정책을 지원한다. 본 연구에서 시뮬레이션한 컴퓨터 시스템 구성은 표 1에 요약되어 있다. MLC STT-MRAM 기반 LLC의 쓰기 동작은 대상 캐시 블록이 어떤 도메인 (소프트 또는 하드)에 사상되는지에 따라 다른 접근 지연시간이 적용되도록 모델링하였다.

성능 평가를 위해 SPEC CPU2006 벤치마크에서 MPKI (Miss Per Kilo Instruction) 값이 큰 메모리 집약적 벤치마크 15개를 사용하였다 또한, 서로 다른 특성을 가진 애플리케이션이 동시에 실행되는 시나리오를 평가하기 위해 서로 다른 SPEC CPU2006 벤치마크로 구성된 MIX 워크로드 10개를 사용하였다.

Table 1. Simulated System Configuration

Number of Cores Processor Clock Issue Width	4 3.2Ghz 8
L1 Cache L2 Cache	32KB, 8-Way, 64B line, 4 Cycles 256KB, 8-Way, 64B line, 12 Cycles
LLC (MLC STT-MRAM) Read Latency (cycles) Write Latency (cycles)	8MB, 16-way, 64B line, 8 Banks Fast: 15, Normal: 22 Fast: 49, Normal: 95
Memory Bus Frequency Memory Channels Ranks per Channel Bank Groups Banks per Bank Group	1600MHz 2 1 4 4
DRAM Access Timing: tRCD-tRP-tCAS tRFC/tREFI	 22-22-22 350ns/7.8us

2. Fast Write Ratio

앞서 설명하였듯이 DBAW 기법에서는 소프트 도메인에 저장된 캐시 블록이 유효하지 않거나 데드 블록 상태면 Fast write을 수행할 수 있다. Fig. 11은 각 SPEC CPU2006 벤치마크를 실행할 때, LLC에 대한 Write request 중 Fast write을 수행한 횟수를 보여준다. 평균적으로 전체 쓰기 요청의 10%에 대해 Fast write을 수행할 수 있는 것으로 확인되었다.

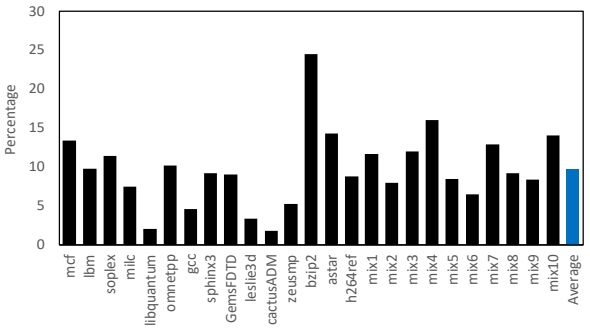


Fig. 11. Percentage of Fast Write

3. Impact on Miss Rate

DBAW 기법을 사용할 경우 소프트 도메인에 저장된 캐시 블록이 데드 블록으로 예측될 경우, 캐시 메모리에서 미리 삭제될 수 있다. 따라서, 데드 블록 예측이 정확하지 않을 경우, 캐시 미스 발생률 (Cache miss rate)을 크게 증가시킬 수 있고 이는 컴퓨터 시스템 성능을 떨어뜨리게 된다. Fig. 12는 CSM 사상 방법을 사용하는 MLC STT-MRAM 캐시 메모리에 DBAW를 적용 시 캐시 미스 발생 증가율을 보여준다. 일부 벤치마크를 제외하고 전반적으로 캐시 미스 발생 증가율이 1% 미만인 것으로 확인되었고, 전체 벤치마크에 대해 캐시 미스 발생률은 평균 0.8% 증가하였다. Bzip2 벤치마크에 대해서는 캐시 미스 발생률이 약 8% 증가하였다. 이는 Fig. 11에서 확인할 수 있는 것과 같이 데드 블록 예측이 잘못된 경우가 많아 Fast write을 과도하게 수행하였기 때문이다. Bzip2와 같은 벤치마크에 대한 데드 블록 예측의 정확도를 개선하면 DBAW 적용에 따른 캐시 미스 발생률 증가를 더욱 감소시킬 수 있다.

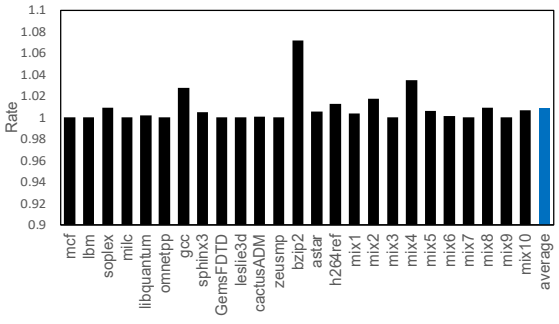


Fig. 12. Impact of DBAW on LLC Miss Rate

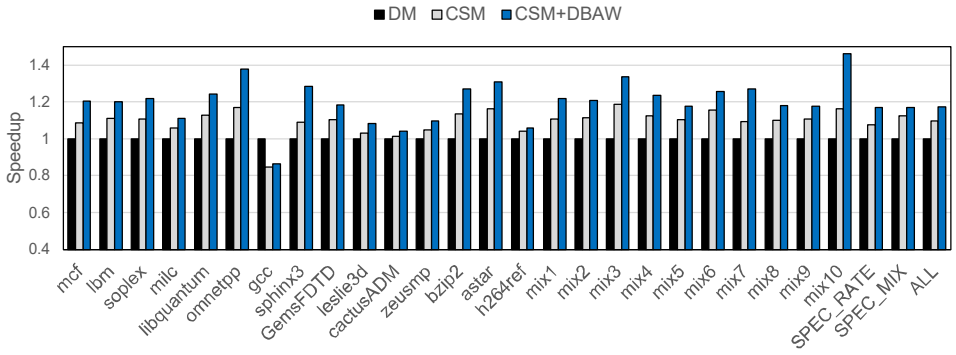


Fig. 13. Performance

4. Performance

Fig. 13은 CSM 방법을 사용하는 MLC STT-MRAM 캐시 메모리에 대해 DBAW 기법 적용 여부에 따른 시스템 성능을 보여준다. CSM 방법은 DM 방법보다 시스템 성능을 평균 10% 향상시킬 수 있는 것으로 평가되었다. CSM 방법에 DBAW 기법을 적용할 때는 7% 추가 성능향상을 얻을 수 있을 것으로 평가되었다. 즉, CSM+DBAW 방법은 DM 방법보다 시스템 성능을 약 17% 향상시킬 수 있다. 앞서 예상한 것과 같이 Fast write을 수행한 비율이 높을수록 DBAW 방법을 사용했을 때의 성능향상이 크다는 것을 확인할 수 있었다.

V. Conclusions

MLC STT-MRAM은 차세대 프로세서의 대용량 온칩 캐시 메모리를 구현할 때 사용될 것으로 기대되고 있다. MLC STT-MRAM은 SRAM 및 SLC STT-MRAM에 비해 높은 데이터 집적도를 제공하지만 쓰기 동작 지연시간이 매우 길다는 단점을 갖고 있다. 본 논문에서는 MLC STT-MRAM 기반 캐시 메모리의 쓰기 동작 성능을 향상시킬 수 있는 매우 효율적인 기법인 데드 블록 탐지 기반 적응형 쓰기 기법 (Dead Block-Aware Adaptive Write Scheme, DBAW)을 제안하였다. 시뮬레이션 기반 성능 평가를 통해 DBAW 기법을 CSM 방법과 함께 사용했을 때 시스템 성능을 평균 17% 향상시킬 수 있음을 확인하였다.

ACKNOWLEDGEMENT

This study was partially supported by the BK21 Plus project (SW Human Resource Development Program for Supporting Smart Life) funded by the Ministry of Education, School of Computer Science and Engineering, Kyungpook National University, Korea (No. 21A20131600005) and by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2019R1G1A1011403).

REFERENCES

[1] Suock Chung et al., "Fully integrated 54nm STT-RAM with the smallest bit cell dimension for high density memory application," Proceedings of 2010 International Electron Devices Meeting, pp. 12.7.1-12.7.4., San Francisco, USA, 2010. DOI: 10.1109/IEDM.2010.5703351

[2] T. Ishigaki, T. Kawahara, R. Takemura, K. Ono, K. Ito, H. Matsuoka, and H. Ohno, "A multi-level-cell spin-transfer torque memory with series-stacked magnetotunnel junctions," Proceedings of 2010 Symposium on VLSI Technology, pp. 47-48, Honolulu, USA, 2010. DOI: 10.1109/VLSIT.2010.5556126

[3] Lei Jiang, Bo Zhao, Youtao Zhang, and Jun Yang. 2012. “Constructing large and fast multi-level cell STT-MRAM based cache for embedded processors,” Proceedings of the 49th Annual Design Automation Conference, pp. 907–912, New York, USA, 2012. DOI: https://doi.org/10.1145/2228360.2228521

[4] Y. Chen, X. Wang, W. Zhu, H. Li, Z. Sun, G. Sun, and Y. Xie, "Access scheme of Multi-Level Cell Spin-Transfer Torque Random Access Memory and its optimization," Proceedings of 53rd IEEE International Midwest Symposium on Circuits and Systems, pp. 1109-1112, Seattle, USA, 2010. DOI: 10.1109/MWSCAS.2010.5548848

[5] X. Bi, M. Mao, D. Wang and H. Li, "Unleashing the potential of MLC STT-RAM caches," Proceedings of 2013 IEEE/ACM

International Conference on Computer-Aided Design (ICCAD), pp. 429-436, San Jose, USA, 2013. DOI: 10.1109/ICCAD.2013.6691153

[6] J. Wang, P. Roy, W. Wong, X. Bi and H. Li, "Optimizing MLC-based STT-RAM caches by dynamic block size reconfiguration," Proceedings of 2014 IEEE 32nd International Conference on Computer Design (ICCD), pp. 133-138, Seoul, South Korea, 2014. DOI: 10.1109/ICCD.2014.6974672

[7] S. Hong, J. Lee and S. Kim, "Ternary cache: Three-valued MLC STT-RAM caches," Proceedings of 2014 IEEE 32nd International Conference on Computer Design (ICCD), pp. 83-89, Seoul, South Korea, 2014. DOI: 10.1109/ICCD.2014.6974666

[8] J. Xu, D. Feng, W. Tong, J. Liu and W. Zhou, "Encoding Separately: An Energy-Efficient Write Scheme for MLC STT-RAM," Proceedings of 2017 IEEE International Conference on Computer Design (ICCD), pp. 581-584, Boston, MA, USA, 2017. DOI: 10.1109/ICCD.2017.100

[9] X. Bi, M. Mao, D. Wang and H. H. Li, "Cross-Layer Optimization for Multilevel Cell STT-RAM Caches," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 6, pp. 1807-1820, June 2017. DOI: 10.1109/TVLSI.2017.2665543

[10] M. A. Qureshi, H. Kim and S. Kim, "A Restore-Free Mode for MLC STT-RAM Caches," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 6, pp. 1465-1469, June 2019. DOI: 10.1109/TVLSI.2019.2899894

[11] S. M. Khan, Y. Tian and D. A. Jiménez, "Sampling Dead Block Prediction for Last-Level Caches," Proceedings of 43rd Annual IEEE/ACM International Symposium on Microarchitecture, pp. 175-186, Atlanta, USA, 2010. DOI: 10.1109/MICRO.2010.24

[12] An-Chow Lai, C. Fide and B. Falsafi, "Dead-block prediction & dead-block correlating prefetchers," Proceedings of 28th Annual International Symposium on Computer Architecture, pp. 144-154, Goteborg, Sweden, 2001. DOI: 10.1109/ISCA.2001.937443

[13] Haiming Liu, Michael Ferdman, Jaehyuk Huh, and Doug Burger, "Cache bursts: A new approach for eliminating dead blocks and increasing cache efficiency," Proceedings of the IEEE/ACM International Symposium on Microarchitecture, pp. 222-233, Los Alamitos, USA, 2008. DOI: 10.1109/MICRO.2008.4771793

[14] M. Kharbutli and Y. Solihin, "Counter-Based Cache Replacement and Bypassing Algorithms," IEEE Transactions on Computers, Vol. 57, No. 4, pp. 433-447, April 2008. DOI: 10.1109/TC.2007.70816

[15] Zhigang Hu, S. Kaxiras and M. Martonosi, "Timekeeping techniques for predicting and optimizing memory behavior," Proceedings of 2003 IEEE International Solid-State Circuits Conference, pp. 166-485, San Francisco, CA, USA, 2003. DOI: 10.1109/ISSCC.2003.1234251

[16] Jaume Abella, Antonio González, Xavier Vera, and Michael F. P. O'Boyle. 2005. IATAC: a smart predictor to turn-off L2 cache lines. ACM Transactions on Architecture and Code Optimization (TACA), Vol. 2, No. 1, pp. 55–77, March 2005. DOI: https://doi.org/10.1145/1061267.1061271

[17] L. Liu, P. Chi, S. Li, Y. Cheng and Y. Xie, "Building energy-efficient multi-level cell STT-RAM caches with data compression," Proceedings of 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 751-756, Chiba, Japan, 2017. DOI: 10.1109/ASPDAC.2017.7858414

[18] N. Chatterjee et al., USIMM: The Utah Simulated Memory Module, tech. report UUCS-12-002, Univ. of Utah, 2012.

Authors

Seokin Hong received the Ph.D. degree in computer science from Korea Advanced Institute of Science and Technology (KAIST), Korea, in 2015. From 2015 to 2017, he had been a senior engineer at Samsung Electronics. In 2017, he moved to IBM T.J. Watson Research Center where he worked on secure processor architectures and emerging memory/storage systems. He is currently an assistant professor at Kyungpook National University. His current research interests include the design of low power, reliable, and high-performance processor architectures and memory systems. He received Best Paper Awards from International Conference on Computer Design (ICCD) in 2010 and Design Automation and Test in Europe (DATE) in 2013.